Java Program Execution: Explained with Compiled, Interpreted, and JIT Compiled Approaches

❖ Compiled Approach:

- The java source code is first compiled using the javac compiler.

this means the code you write in a java file is not directly understood by the computer. So, it needs to be converted into another form using the javac compiler. The compiler translates it into bytecode, which the JVM can understand.

- This generates a bytecode file, which is platform-independent.

The compiler creates a .class file that contains bytecode. **Bytecode** is an intermediate language between java source code and machine code. It is a special format that is not tied to any operating system. It can run on any device that has a Java Virtual Machine (JVM), no matter if it's Windows, Mac.

- The bytecode is then executed by the java virtual machine **(JVM).**

After the .class file is created, it is run by the Java Virtual Machine (JVM). The JVM reads the bytecode and either interprets it line by line or compiles it into machine code using JIT (Just-In-Time compiler). This is what makes Java programs run on any device that has a JVM — without rewriting the code.

❖ Interpreted Approach:

- The Java Virtual Machine (JVM) reads the .class file and executes the bytecode one instruction at a time. It does not convert the whole program into machine code in advance. Instead, it interprets and runs the code as it goes, which may make execution a bit slower than fully compiled languages.

JIT Compiler:

- The JVM uses a Just-In-Time (JIT) compiler to improve performance during execution. The JIT compiler is a part of the JVM that detects frequently used bytecode and converts it into machine code while the program is running, instead of interpreting it every time.